Development of R-Cube Manipulation Language

-Accessing Real Worlds Over the Network-

Yasuyuki YANAGIDA, Naoki KAWAKAMI, Susumu TACHI

School of Engineering, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113 Japan
{yanagida, kawakami, tachi}@star.t.u-tokyo.ac.jp

Abstract

R-Cube (Real-time Remote Robotics: R^3) is the concept which enables human beings to exist anywhere in the world by controlling remote robots in real time through the network. So far there exists a standardized method to describe and access virtual worlds by using Virtual Reality Modeling Language (VRML). However, accessing real environments is not considered by the current version of VRML. On the other hand, several experiments to control remote robots over ISDN and/or Internet are reported, but we do not have a standard method to access real environments yet. In order to provide a method to access real worlds over the network easily, we propose a language RCML (R-Cube Manipulation Language) to describe remote robots and environments, and a protocol RCTP (R-Cube Transfer Protocol) to communicate with the remote robot site, based on the requirements for real-time remote robotics. The basic concept of RCML/RCTP systems, the design of the functional configuration of the system, and an experimental implementation of the server/client software are also shown.

Key words: Tele-existence, R-Cube, RCML, RCTP, VRML, Java

1. Introduction

The concept of R-Cube (Real-time Remote Robotics: R^3) shows us the possibility that human being can *tele-exist* anywhere in the world with a sensation of presence, feeling and acting as if he/she really existed in the remote environment, by controlling remote robots over the network[1-3]. So far the information exchanged over the network has been captured within the world consisting of the network itself and computers, separated from the real world which we actually live in. R-Cube is considered to be an attempt to release such information from such closed space so that it acts as the medium of actual behavior. When the concept of R-Cube comes true, we can control the remote robots located in offices, public spaces, places which are too dangerous for human beings to visit, etc., from control terminals located in safe and comfortable places, such as homes, another offices, public "R-Cube stations", and so on (Figure 1). Thus we will be able to exist and act freely in the remote environment, regardless of physical limitation of time and space.

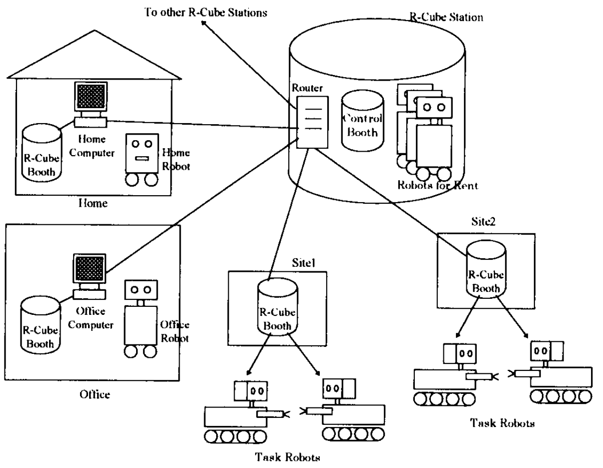

Fig. 1 Concept of R-Cube Network.

Towards the realization of this concept, we approach from the end-user's stand point, focusing on "how the users access the remote *real* environments". Our goal is to provide an intuitive and easy method to access remote environments by controlling remote robots using the user's computer connected to the network.

2. Related Works

The use of Virtual Reality Modeling Language (VRML) [4,5] has become the most popular method to describe and access *virtual* environments. The recent progress of VRML provides the feature that multiple users can share the world (i.e., exist in the same virtual environment simultaneously) and interact with each other [5,6]. Another good news of VRML for users who want to navigate and act in virtual worlds is that it was announced as ISO standard, VRML97. This provides the users stable circumstances to access a number of virtual worlds.

However, accessing *real* worlds is not considered in the current version of VRML, i.e., the users cannot go out of the world consisting of computers and networks. To enable ourselves to access real worlds, it is necessary to incorporate a mechanism of controlling remote robots into VRML worlds.

On the other hand, several experiments to control actual robots over ISDN and/or Internet are reported, and some of them are opened to everyone in the world through the Internet[7,8]. However, there is no standard method to control these robots, except for the limited user interface provided by WWW browser.

Conventional "Internet controlled robots" use standard Hyper-Text Markup Language (HTML) and Hyper Text Transfer Protocol (HTTP). This approach has an advantage that users can control robots by using general WWW browser. However, this also implies the following problems:

- The kinds of user interface are limited to those supported by standard HTML language. It is not sufficient for our purpose to limit ourselves to use only conventional graphical user interface (GUI), as the use of various input devices such as three dimensional position/orientation sensors are more intuitive to control the robots.
- Using HTML, the information exchanged over the network is symbolic information attached to correspondent GUI manipulation, rather than control information itself. This is not suitable for continuously controlling robots by specifying numerical parameters such as hand position, head orientation, etc., though it may be sufficient for controlling robots by symbolic commands.
- The feature of the HTTP protocol (version 1.0), that is, the connection between the server and client software is disconnected after sending/receiving the content of document, is not suitable for continuous control of remote robots.

These problems are inherent as far as the system is based on standard HTML and HTTP, and an effort to decrease such disadvantages within the frame of existent language and protocol may result in constrained, not intuitive systems.

To overcome these problems, we propose a new description language RCML (R-Cube Manipulation Language) and a protocol R-Cube Transfer Protocol (RCTP) for controlling remote robots.

3. Application Image Using RCML

Speaking of accessing three-dimensional environments, we already have a standard method for accessing *virtual* environment, by using VRML[4,5] browser. From the view point of end users, the application aspect of R-Cube and that of using VRML browser differs only in that the operator accesses whether *real* or *virtual* environment. In order to smoothly introduce R-Cube application to net-

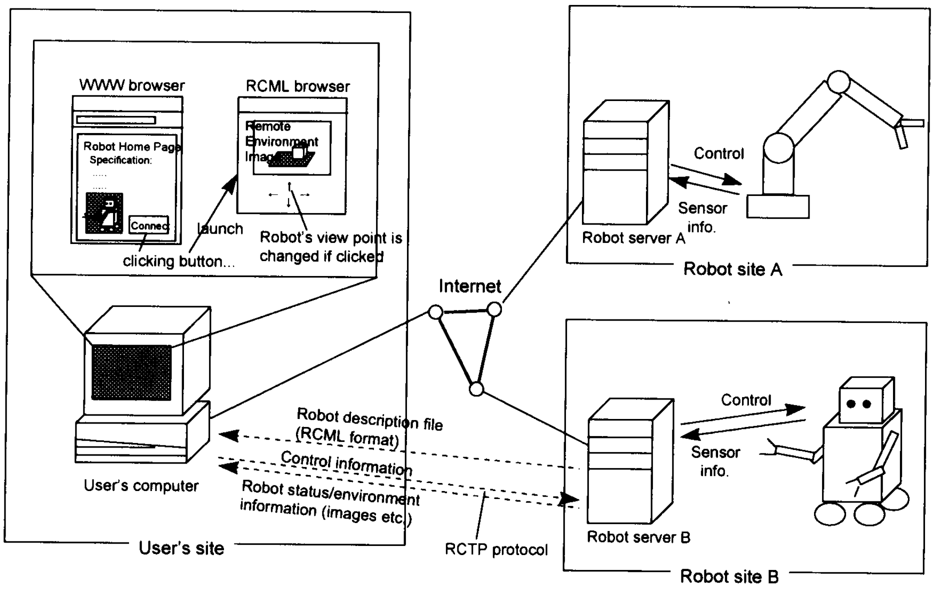

Fig.2 Application Image of RCML system

work users, it is necessary to design a method of accessing real worlds just in the same manner as accessing virtual worlds, that is, we have to establish seamless access to real and/or virtual environments.

Now let us consider how we access virtual worlds using VRML. The description file written in VRML format is downloaded from WWW server, and then the VRML browser software is invoked to navigate and make actions in the virtual environment. We are going to design a method which the users regard just as the same way as using VRML browser, to provide easy and convenient access to real worlds.

The concept of RCML application is shown in Fig. 2. The user accesses the Web page describing the information of the robot in the form of hyper-text and graphics, using WWW browser. If the user clicks the icon, the description file written in RCML format is downloaded to the user's computer and the RCML browser is launched. The RCML browser parses the downloaded file to know the geometry information including the alignment of the degrees-of-freedom, controllable parameters and their limitation, available sensory information, etc. It then decides what kind and number of devices are required to control the remote robot, and then it generates a GUI panel used for controlling the remote robot, a video window to display video image of remote environment, and "monitoring window" to watch the status of remote environment including the robot. If the user has devices such as 6DOF position/orientation sensors, the user can indicate the browser software to use those devices instead of conventional GUI interfaces.

4. Basic Design

To satisfy the above requirements, we propose a new language "R-Cube Manipulation Language" (RCML), which describes the feature of remote robots and environments, and a communication protocol "R-Cube Transfer Protocol" (RCTP), which is designed to exchange the control and sensory information between the robot site and the user site.

4.1 System Configuration

The functional system configuration is shown in Fig. 3.

4.1.1 Server/Client Software

The RCML/RCTP system consists of server and client software. The server stands for the software running at the site where the robot exists, and the client stands for the software running at the user's site.

4.1.2 Input and Output Devices

Here we call *device* a functional unit to intermediate the *outside* (real) world and the *inside* world (information space: computer and network side). Referring to the direction, the *input* device is the one which brings physical information of the real world into the information space, and the *output* devices as their counterpart. For example, position/orientation sensor to measure the motion of the operator is categorized to input device on the client site, and the motor controller of the robot is categorized to output device on the server site. Sometimes a single block of physical devices is considered to have both input and output devices, e.g., most robot arms have motor controller units as output device and joint angle sensors as input device.

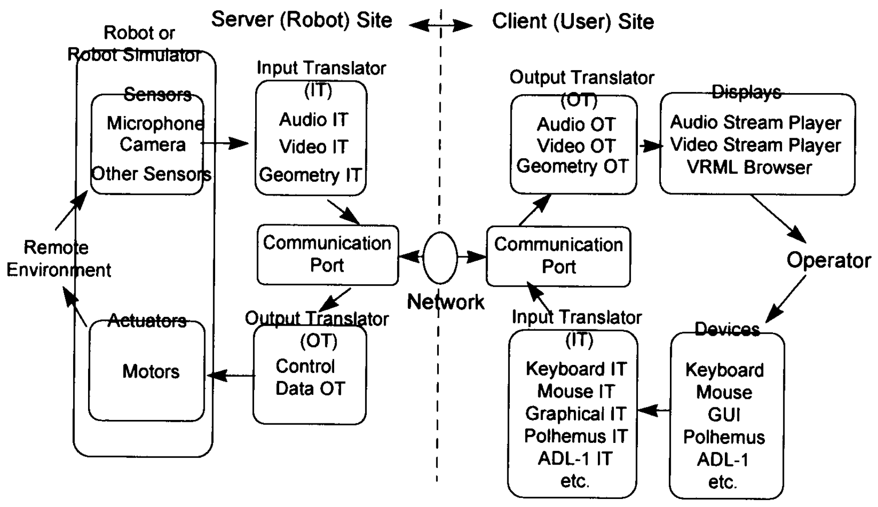

Fig. 3 Functional Configuration of RCML/RCTP System

4.1.3 Translators

In order that the RCML/RCTP system is applicable for various types of robots and devices, the data transferred over the network should be independent of the robot and/or device connected at each site. To satisfy this requirement, software modules called *translators* are provided to intermediate robots/devices and the network.

4.1.4 Modules and Signal/Information Flow

The modules mentioned above are classified in the sense of their function and are independent of physical units. For example, an 6DOF geometry input device (such as ADL-1 or Polhemus Tracker) is treated as a device which provides 6DOF numerical value. This means that the module called "device" in RCML/RCTP system can include physical sensor unit, signal processing unit, local communication channel (such as RS-232), and the device driver software installed in the computer.

Any signal in the real world is captured by the input device at client (server) site, converted to the standard form through the input translator, sent to server (client) site through the communication port, converted to the device dependent form through the output translator, and then put out of the computer through the output device. Thus, an input device, an input translator, an output translator, and an output device compose a set of modules for one signal/information flow.

4.2 Logical Objects

In the RCML/RCTP system, each pair of device and translator composes a logical *object*, which is used as a unit of sending and receiving commands, data and messages. Currently the kinds of logical object are as follows:

Input device objects:
- Video Input
- Audio Input
- Control Input
- String Input

Output device objects:
- Video Output
- Audio Output
- Control Output
- String Output

Video Input object is used to get live video stream, and Audio Input object is used to get live audio stream. Control Input object is used to get control data of various degrees-of-freedom, such as 6 DOF position/orientation sensors, 2D or 3D mice, and so on. Also conventional GUI can be used for control input object. String Input object is used to get arbitrary kind of symbolic information including character string, such as command string used by command-based robot control.

Each category of output object is the counterpart of corresponding category of input object.

In addition, several kinds of object are provided to control the entire software system.

System Objects:
- System
- Server
- Client
- Onlooker

System objects stand for the manager of the whole software system. Only one client object can exist in the whole system to avoid the conflict when multiple users try to control a single robot simultaneously.

4.3 RCML

Based on the design mentioned above, the language RCML is used for integrated description of the information of remote robot control and the information of remote environment. RCML covers the following information:

- Geometry of the robot and the configuration including degrees of freedom, workspace, controllable parameters, and so on.
- Kinds and specification of sensory information available from the robot including video (by robot cameras), sound (by microphones), range data, and so on.
- Remote environment information.

Among these kinds of information, the world wide standard VRML can be used for geometry description and environment modeling. It is not a good way to describe such information in the form of other original language, because it is hardly considered that users select a new format of description incompatible with the existent standard. Fortunately, VRML is flexibly designed to be easily extended. For this reason, we decided to describe the above information in the form of VRML node extension, so that the user can use both RCML-compliant browser and ordinary (not RCML-compliant) browser. RCML-related part of the description can be neglected by browser software which does not understand RCML extensions. The actual specification of RCML is still under construction.

4.4 RCTP

The protocol RCTP is used for communication between the server (robot) site and the client (user) site. It covers the following kinds of information:

- System information: includes connect/disconnect requests, error status, etc.
- Control information: The value obtained from user's input devices, such as 6DOF position/orientation sensors, GUI control information, keyboard and/or mouse inputs, and so on.
- Sensory information: The sensory data displayed to the user can be transmitted, though one can send such information using separate connection.

First, when the client requests for connection with the server, the server checks if other client is occupying the robot. If so, the client software is rejected to controlling the robot, entering "onlooker" mode. Otherwise, the client is permitted to control the robot. This procedure is called authorization phase.

Next, the system enters the negotiation phase. In this phase, the client assigns controllable objects at the remote site with the available devices, as well as assigning remote sensory information channel with the local output devices. For example, if the server site has a Control Output object with 6 DOF, the client attempts to assign a Control Input object with 6 DOF. Similarly, if the server site has a Video Input object, the client attempts to assign a Video Output object. This assignment procedure may be regarded as "making the copy of the objects at remote site", or "making the pair of input and output objects".

As described in Section 3, we provide an "monitoring window", which displays the image of virtual space as a copy of the remote environment. In this case, this monitoring window is handled as a control output object, not as a video output object.

After finishing the negotiation phase, the system enters actual data communication phase. The communication is taken place based on the object oriented manner. While connection is established, the control data transferred over the network must be in generic form, which is independent of specific devices, so that any kind of robot with various DOF and geometry can be controlled using this system.

5. Experimental Implementation

To demonstrate the concept of RCML/RCTP, we implemented an experimental system. We provided the following objects as the controlled target:

- Controllable camera
- Virtual robot

"Controllable camera" is a camera whose orientation (yaw and pitch: 2 DOF) is controllable by computers through RS-232C is equipped. This is an representation of the robot head equipped with cameras, and is considered to be one of the simplest example of the controllable object. The available information/signals from the target are 2 DOF orientation status and the video signal.

Also a "virtual robot" which has 14 DOF (3 for head, 7 for right arm, 1 for hand grasp, 2 for horizontal position, and 1 for orientation) controllable parameters is implemented on the server computer, to demonstrate a more complicated controllable target. The kind of available information/signals are the current status of the above controllable parameters and the video image generated by the virtual camera equipped on the robot head. This example also demonstrates the role of *translators*. The robot has a 7 DOF arm including 1 redundant DOF, though the general position/orientation sensor devices available at the user's site has only 6 DOF (3 for position and 3 for orientation). This implies the necessity of solving inverse kinematics before sending control output to the robot controller. In this case, the module for solving inverse kinematics is provided at the server's output translator.

We used the existing software as well as possible, resulting in less amount of program coding. The server software is implemented on Silicon Graphics workstations based on the virtual robot simulator[9] written in C++ language, and the client software is implemented on PCs running Windows95 using Java. Community Place Browser from Sony is used as the VRML browser. Currently sensory signals such as video and audio are transferred from the server site to the client site through independent connection, by using existing tools such as RealVideo server and player software.

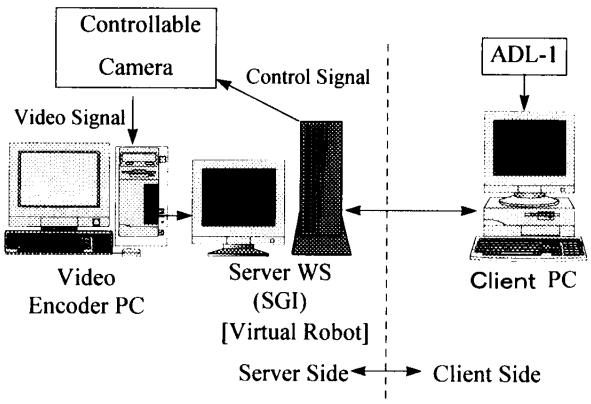

Fig. 4 Configuration of the Experimental System

6. Conclusion

A set of description language RCML and protocol RCTP is proposed to enable easy and comprehensive access to real environments. The conceptual design and the implementation is still under progress so that we are ready to try better solutions.

Acknowledgment

The authors would like to thank K. Matsuda and F. Hiei at Sony Architecture Laboratory for many useful advice.

References

1. MITI of Japan, R-Cube WG ed.: "R-Cube", Nikkan Kogyo Shimbun (1996)
2. S. Tachi: "Virtual Reality and Robotics", Journal of the Rbotics Society of Japan, Vol. 15, No.4, pp.32-35 (1997)
3. http://www.irofa.com/rcube/
4. M. Pesce: "VRML Browsing and Building Cyberspace", Macmillan Publishing (1995)
5. K. Matsuda, et al.: "Virtual Society: Multi-use Interactive Shared Space on WWW", Proc. of the 6th International Conference on Artificial Reality and Tele-Existence (ICAT '96), pp. 83-95, Makuhari, Japan (1996)
6. http://vs.sony.co.jp/
7. http://www.usc.edu/dept/raiders/story/index.html
8. http://www.cs.cmu.edu/~xavier/
9. Y. Yanagida and S. Tachi: "Virtual Environment Construction Method Using Class Library", The Transactions of The Institute of Electrical Engineers of Japan, Vol. 115-C, No. 2, pp.236-244 (1995)